

Introduction to Timing Exceptions Demo Script

Introduction

This demonstration provides instructions for creating multicycle path and false path constraints using Tcl commands.

Preparation:

- Required files: \$TRAINING_PATH/TimingExceptions_Intro/demo/KCU105/netlist
- Required hardware: None
- Supporting materials: *wave_gen* detailed design description document

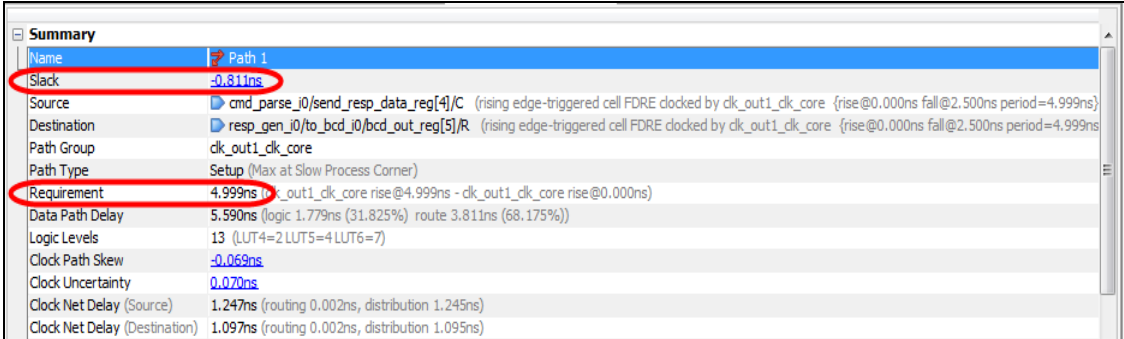
Timing Exceptions

| Action with Description | Point of Emphasis and Key Takeaway |
|---|--|
| <ul style="list-style-type: none"> • Launch the Vivado™ Design Suite. • Unzip the project using the Tcl Console: <pre>exec unzip \$::env(TRAINING_PATH) / TimingExceptions_Intro/demo/ KCU105/netlist.zip -d \$::env(TRAINING_PATH) / TimingExceptions_Intro/demo/ KCU105/netlist</pre> | <ul style="list-style-type: none"> • You can easily open the project from the Quick Start page. |
| <ul style="list-style-type: none"> • Open the wave_gen.xpr project located from the following directory: \$TRAINING_PATH/TimingExceptions_Intro/demo/KCU105/netlist | |
| <ul style="list-style-type: none"> • Open the wave_gen_timing.xdc file and view the constraints that are present. | <ul style="list-style-type: none"> • What are all the constraints present in the XDC file? <ul style="list-style-type: none"> • <i>wave_gen_timing.xdc</i> contains: <ul style="list-style-type: none"> ▪ create_clock ▪ set_input_jitter ▪ create_generated_clock ▪ set_input_delay ▪ set_output_delay ▪ set_clock_groups |

| Action with Description | Point of Emphasis and Key Takeaway |
|--|--|
| <ul style="list-style-type: none"> Open the implemented design and view the Timing Summary Report. | <ul style="list-style-type: none"> You can open the implemented design by using either: <ul style="list-style-type: none"> Flow Navigator Tcl Console Horizontal toolbar The Timing Summary Report is generated automatically when the implemented design is opened. |
| <ul style="list-style-type: none"> What are timing exceptions? <ul style="list-style-type: none"> Timing exceptions modify the existing setup/hold check. They provide the designer with the ability to accurately describe "non-default" timing relationships on static timing paths. | |
| <ul style="list-style-type: none"> What are the paths that are failing in the design? | <ul style="list-style-type: none"> The paths from rst_pin are also failing, which can be seen under the Other Path Groups section. |
| <ul style="list-style-type: none"> View the <i>wave_gen</i> detailed design description document provided in the <i>TimingExceptions_Intro</i> directory to learn more about the <i>wave_gen</i> design. | <ul style="list-style-type: none"> Read the Response Generator section (page 12) for more details. |

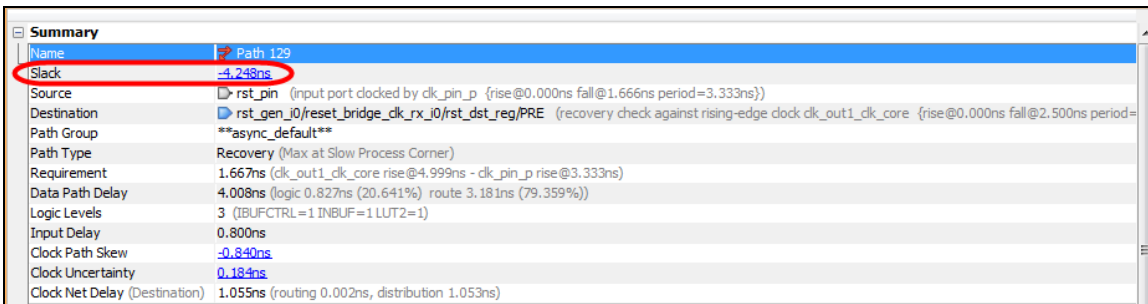
Multicycle Path

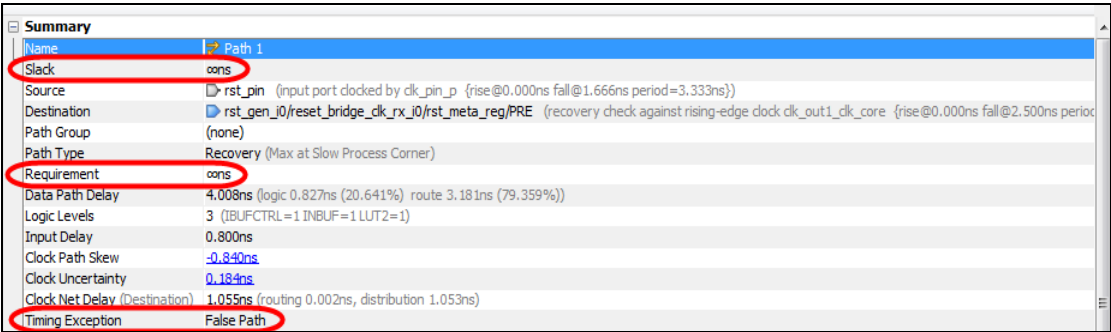
| Action with Description | Point of Emphasis and Key Takeaway |
|--|--|
| <ul style="list-style-type: none"> Select the failing path with the worst slack from Intra-Clocks Paths section and double-click the path to open the path timing report. | <ul style="list-style-type: none"> Note the Requirement and Delay values from the report. After some research and debugging, you can find out that this path is a multicycle path and needs a timing exception constraint. |

| Action with Description | Point of Emphasis and Key Takeaway |
|--|---|
|  <p>Note: The timing numbers may vary depending on the version of the Vivado Design Suite and the OS.</p> | |
| <ul style="list-style-type: none"> Enter the following command in the Tcl Console: <pre>set_multicycle_path -from [get_cells {cmd_parse_i0/send_resp_data_reg [*]]} -to [get_cells {resp_gen_i0/to_bcd_i0/bcd_out_r eg[*]]} 2</pre> This creates a multicycle constraint from cmd_parse_i0/send_resp_data_reg[*] to resp_gen_i0/to_bcd_i0/bcd_out_reg[*] by a factor of two. | <ul style="list-style-type: none"> Multicycle paths occur when registers are not updated on consecutive clock cycles <ul style="list-style-type: none"> Always at least N clock cycles between the time the source flip-flops update and the destination flip-flops capture the result. Syntax: <ul style="list-style-type: none"> set_multicycle_path <path_enumeration> <check_type> \<multiplier> <path_enumeration> are options to identify the set of paths to which to apply the set_multicycle_path command. <ul style="list-style-type: none"> Uses the -from, -through, and -to options (among others). <check_type> determines the static timing check affected by -setup or -hold. <ul style="list-style-type: none"> When neither -setup nor -hold is used, the path multiplier applies to both setup and hold checks. <multiplier> determines how many clock periods to use for the multicycle path. |
| <ul style="list-style-type: none"> Rerun the Timing Summary report. | <ul style="list-style-type: none"> Did the multicycle setup path constraint resolve timing failures on the path? <ul style="list-style-type: none"> No, the timing now fails on the hold check of the path in clk_out1_clk_core. |

| Action with Description | Point of Emphasis and Key Takeaway |
|--|---|
| <ul style="list-style-type: none"> Why did the hold check fail? <ul style="list-style-type: none"> By default the hold capture edge is one clock period before the setup capture edge. This is generally not the correct hold relationship to check for a multicycle path. The hold capture edge usually needs to be modified with the multicycle hold constraint. | |
| <ul style="list-style-type: none"> Enter the following command in the Tcl Console to create a multicycle hold path: <pre>set_multicycle_path -from [get_cells cmd_parse_i0/send_resp_data_reg[*]] -to [get_cells resp_gen_i0/to_bcd_i0/bcd_out_re g[*]] -hold 1</pre> | <ul style="list-style-type: none"> Syntax: <code>set_multicycle_path -from \$from_list -to \$to_list -hold <N-1></code> <ul style="list-style-type: none"> Returns the hold capture edge to the correct edge. |
| <ul style="list-style-type: none"> Rerun the Timing Summary report. | <ul style="list-style-type: none"> This loads the new timing values. |
| <ul style="list-style-type: none"> Enter the following command in the Tcl Console to view the path timing report of the constrained path. <pre>report_timing -from [get_pins {cmd_parse_i0/send_resp_data_reg [4]/C}] -to [get_pins {resp_gen_i0/to_bcd_i0/bcd_out_r eg[5]/D}] -name timing_2</pre> Double-click the path to open the path timing report. | <ul style="list-style-type: none"> Note the requirement and delay values. |
| <p>The screenshot shows the 'Summary' window in Vivado. The 'Path 1' is selected. The 'Slack' is 4.544ns. The 'Requirement' is 9.999ns. The 'Timing Exception' is 'MultiCycle Path Setup -end 2'. Other values include: Source: cmd_parse_i0/send_resp_data_reg[4]/C, Destination: resp_gen_i0/to_bcd_i0/bcd_out_reg[5]/D, Path Group: clk_out1_clk_core, Path Type: Setup (Max at Slow Process Corner), Requirement: 9.999ns, Data Path Delay: 5.378ns, Logic Levels: 13, Clock Path Skew: -0.069ns, Clock Uncertainty: 0.070ns, Clock Net Delay (Source): 1.247ns, Clock Net Delay (Destination): 1.097ns.</p> | |
| <p>Note: The timing numbers may vary depending on the version of the Vivado Design Suite and the OS.</p> | |
| <ul style="list-style-type: none"> What is the value of the requirement and delays now? <ul style="list-style-type: none"> Due to the <code>set_multicycle_path</code> command, the requirement for this path has changed from 4.999 to 9.999 ns; i.e., double the clock period. The source clock and datapath delays are the same as a single-cycle path. | |

False Path

| Action with Description | Point of Emphasis and Key Takeaway |
|--|---|
| <ul style="list-style-type: none"> What are false paths? <ul style="list-style-type: none"> Setup and hold checks are all performed on the static timing path. There are circumstances where these checks are not valid. A false path constraint disables the setup and/or hold check on paths. For example, the rst_pin input can be safely considered asynchronous to the design since it is synchronized to each of the different clock domains in the design using an instantiation of the reset_bridge module. As a result, the paths from the rst_pin input are false paths. | |
| <ul style="list-style-type: none"> Select the path from the Timing Summary report that is failing (paths from the rst pin are failing). Double-click the selected path to open the path timing report and note the requirement on the path. | <ul style="list-style-type: none"> Note the requirement and slack values. |
|  <p>Note: The timing numbers may vary depending on the version of the Vivado Design Suite and the OS.</p> | |
| <ul style="list-style-type: none"> Enter the following command in the Tcl Console: <pre>set_false_path -from [get_ports rst_pin]</pre> | <ul style="list-style-type: none"> Syntax: <code>set_false_path <path_enumeration> <check_type></code> |
| <ul style="list-style-type: none"> Re-run and review the Timing Summary report. | <ul style="list-style-type: none"> Are there any timing failures in the design? <ul style="list-style-type: none"> There are no timing failures in the design. |
| <ul style="list-style-type: none"> Check the timing path report on the rst_pin in the Timing Summary report. | <ul style="list-style-type: none"> False paths will not show up in any timing report. |

| Action with Description | Point of Emphasis and Key Takeaway |
|---|---|
| <ul style="list-style-type: none"> Enter the following command in the Tcl Console: <pre>report_timing -from [get_ports rst_pin] -to [get_pins rst_gen_i0/reset_bridge_clk_rx_i0/rst_meta_reg/PRE] -name timing_3</pre> Double-click the path to view the path report. | <ul style="list-style-type: none"> If a <code>report_timing</code> command specifically enumerates a false path, then path properties on a false path can be viewed. If explicitly enumerated, the slack will be reported as inf. |
|  <p>Note: The timing numbers may vary depending on the version of the Vivado Design Suite and the OS.</p> | |
| <ul style="list-style-type: none"> Save the constraints, re-run the implementation, and check the timing report. Reload the implemented design. Close the implemented design and exit the Vivado Design Suite. | |

Summary

This demonstration illustrated how to create multicycle and false path constraints using Tcl commands. You also learned how to view the path timing report for specific paths. In the lab, you will learn to apply these constraints using the GUI.

References:

- Supporting materials
 - Vivado Design Suite User Guide: Using Constraints* (UG903)